[redacted]

ECE 331
020718
↳ 3:30p

Homework #3

) 2.3-7 (pg 39)

"Describe a $\Theta(n\lg n)$-time algorithm that, given a set S of n integers and another integer x, determines whether or not there exist two elements in S whose sum is exactly x."

*NOTE: when seeing $\Theta(n\lg n)$, think of "divide-and-conquer" algorithms.

First, we want to sort the set S using MERGE-SORT to get it ascending in order. By using for loops that run "n" times, we can apply a BINARY search to every $S[i]$. We have a worse case of $\Theta(n\lg n)$ for the MERGE and $\Theta(\lg n)$ for BINARY. Finally, because it runs n times, the total time is $n * \Theta(\lg n) = \Theta(n\lg n)$.

Polynomial Evaluation

3 Horner's Rule

$P(x) = \sum_{k=0}^{n} a_k x^k = a_0 + x(a_1 + x(a_2 + \dots + x(a_{n-1} + x a_n)\dots))$

a) "In terms of θ-notation, what is the asymptotic running time of this code?"

The algorithm runs in a loop for all elements, n, and the time taken, θ. Therefore, the run time is $\boxed{\theta n}$

b) NAIVE-POLY-EVAL(A, x) // A is an array

```
y = 0
for i = 1 to A.length
    m = 1
    for j = 1 to i-1
        m = m * x
    y = y + A[i] * m
```

Here, we have *two* for loops, each running at n times. Therefore, we'd have θn^2, which is <u>slower</u> than Horner's Rule.